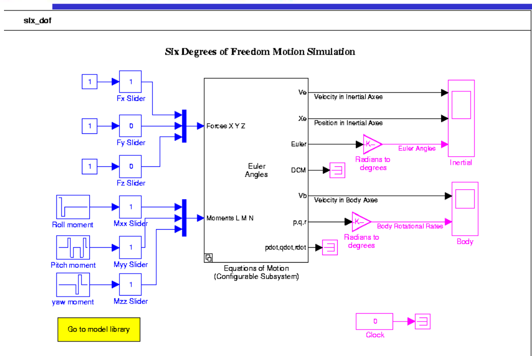
# Program Synthesis: State Estimation

Dr. Michael R. Lowry
ASE Group
NASA Ames

# Code Generation: now & future

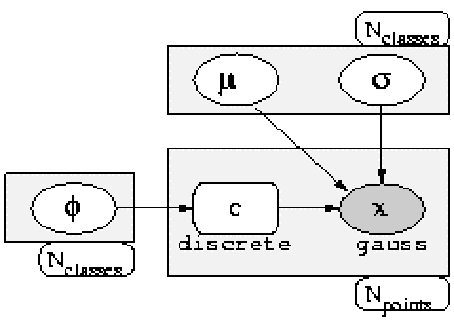



simulink model

generate code

stub code

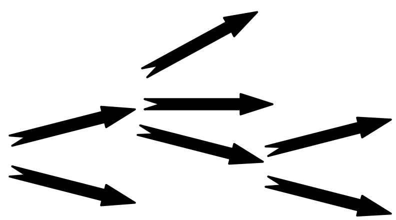
is it what I want?




autofilter
model

model specific
optimization

domain
knowledge



```
gain := pminus *
mtrans(h) *
minv(h * pminus *
mtrans(h) +
r);
xhat1 := xhatmin
+ gain * (zhat -
zpred);
```

explicit assumptions

certified code

synthesize code

multiple programs

# Autofilter Spec Language

State variables

Process noise variables

Process model

$$\dot{\mathbf{x}} = \mathbf{F}\mathbf{x} + \mathbf{G}\mathbf{u}$$

Measurement model

$$\mathbf{z} = \mathbf{H}\mathbf{x} + \mathbf{v}$$

Measurement variables

Measurement noise variables

$$+$$

Filter architecture

Sensor specs

variable

- units
- frames
- coordinate systems
- distributions

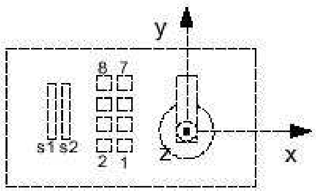
# FY02 State Estimation Synthesis

- Major FY02 study focusing on DS-1 ACS. With JPL avionics branch, specified core components of ACS, synthesized code, and benchmarking synthesized code against manually developed ACS (9/25/02) on autonomy testbed
- FY02: Synthesis capability for state estimation code that is robust in presense of sensor failures.
  - Rover with wheel sensors and DG
  - Instrument failures in aviation
  - Space vehicle docking thruster control

# State Estimation Synthesis FY02

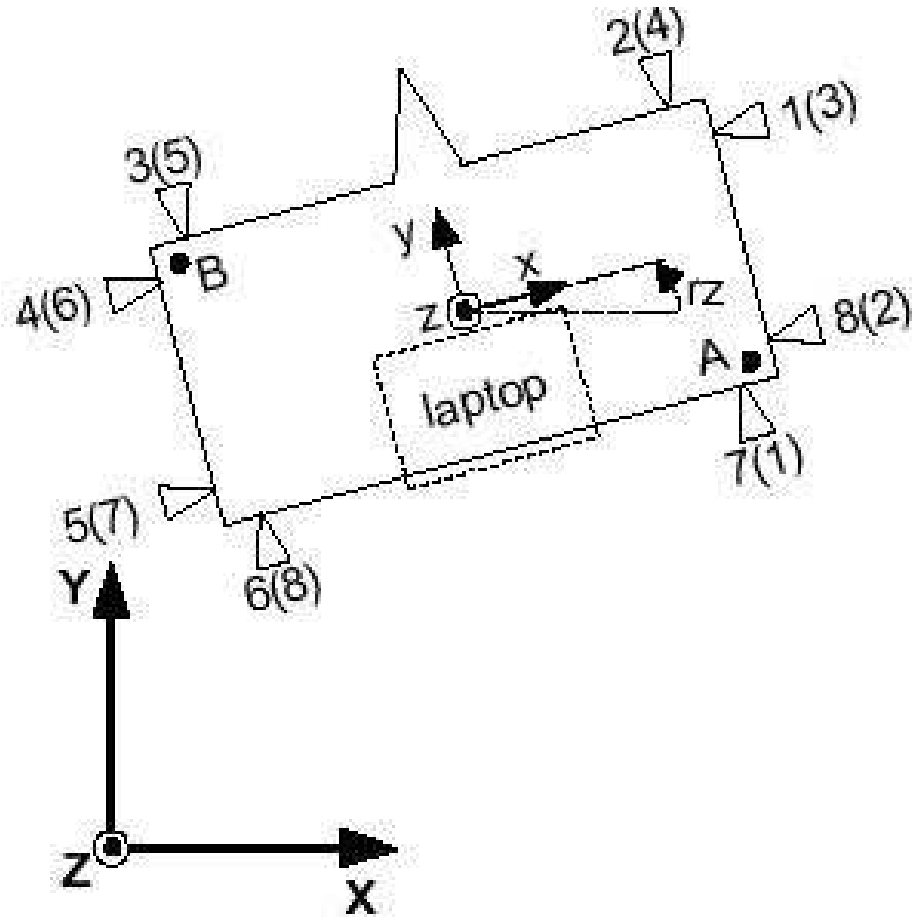
- Filter capabilities
  - Kalman filter (linear model)
  - Linearized Kalman filter (e.g. INS aided navigation)
  - Extended Kalman filter (e.g., within GPS)
  - Unscented Kalman filter
  - Particle filter
- Process model
  - Specifications from differential (continuous) or difference (discrete) equations
  - Symbolic analysis system implemented
- Sensor Modeling
  - Guassian
  - Failure modes

# Smart Systems Spacecraft Simulator




BG Systems Joystick

**Dynasights**
A - bottom, port#3
B - top, port#4





Ed Wilson

3/23/01

# Kalman Filter Design I

force.vframe = $f2f \cdot T$

accel.CMframe = $f2a \cdot \cancel{T_{xm}} \cdot$ force.vframe — $T_{xm}$ handles off-center CM

accel.vframe = accel.CMframe + $\cancel{CM3Z \times}$ — for now, assume CM is centered

accel.iframe = $T_x X \cdot$ accel.vframe — $T_x X$ rotates from vehicle to inertial

so, accel.iframe = $T_x X \cdot f2a \cdot f2f \cdot T$

frames

$$T_x X = \begin{bmatrix} \cos(rz) & -\sin(rz) & 0 \\ \sin(rz) & \cos(rz) & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad f2a = \begin{bmatrix} 1/m & 0 & 0 \\ 0 & 1/m & 0 \\ 0 & 0 & 1/I \end{bmatrix}$$

$$f2f = \begin{bmatrix} -F & 0 & 0 & F & F & 0 & 0 & -F \\ 0 & -F & -F & 0 & 0 & F & F & 0 \\ F_s & -F_l & F_l & -F_s & F_s & -F_l & F_l & -F_s \end{bmatrix}$$

F: Force per thruster (N)
Fs: F × short moment arm (N-m)
Fl: F × long " " "

process noise

state variables

$$\begin{bmatrix} p.x.i(k+1) \\ p.y.i(k+1) \\ p.rz.i(k+1) \\ v.x.i(k+1) \\ v.y.i(k+1) \\ v.rz.i(k+1) \end{bmatrix} = \begin{bmatrix} 1 & & & T_s & & \\ & 1 & & & T_s & \\ & & 1 & & & T_s \\ & & & 1 & & \\ & & & & 1 & \\ & & & & & 1 \end{bmatrix} \cdot \begin{bmatrix} p.x.i(k) \\ p.y.i(k) \\ p.rz.i(k) \\ v.x.i(k) \\ v.y.i(k) \\ v.rz.i(k+1) \end{bmatrix} + \begin{bmatrix} T_{s2} & & \\ & T_{s2} & \\ & & T_{s2} \\ T_s & & \\ & T_s & \\ & & T_s \end{bmatrix} \cdot \begin{bmatrix} a.x.i(k) \\ a.y.i(k) \\ a.rz.i(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ w.x(k) \\ w.y(k) \\ w.rz(k) \end{bmatrix}$$

$$X(k+1) = \underset{G}{A}\; X(k) + \underset{H}{B}\; u(k) + w(k)$$

$$T_{s2} = \frac{T_s^2}{2}$$

# Kalman Filter Design II

$$\begin{bmatrix} os.x.A.cal \\ y\,A \\ x\,B \\ y\,B \end{bmatrix} = \begin{bmatrix} p.x.i \\ p.y.i \\ p.x.i \\ p.y.i \end{bmatrix} + \begin{bmatrix} l_xA\cdot\cos - l_yA\cdot\sin \\ l_xA\cdot\sin + l_yA\cdot\cos \\ l_xB\cdot\cos - l_yB\cdot\sin \\ -l_xB\cdot\sin + l_yB\cdot\cos \end{bmatrix}$$

cos is $\cos(p.rz.i(k))$
sin is $\sin(\quad)$

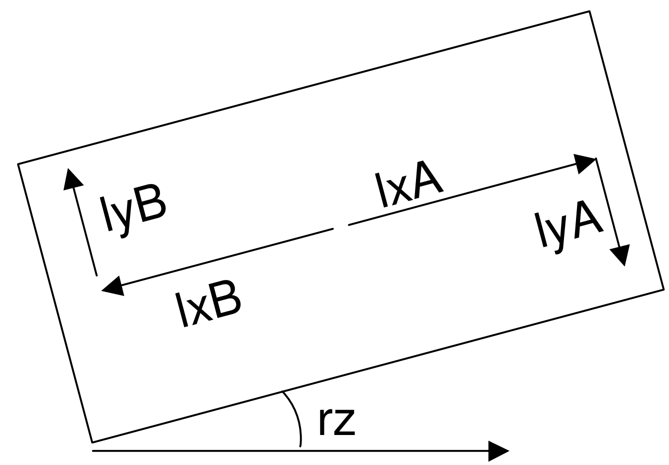
$$ROT = \begin{bmatrix} \cos & -\sin \\ \sin & \cos \end{bmatrix}$$

$$\begin{bmatrix} \\ \\ \\ \\ \end{bmatrix} = \begin{bmatrix} \\ \\ \\ \\ \end{bmatrix} + \begin{bmatrix} ROT & 0 \\ 0 & ROT \end{bmatrix} \cdot \begin{bmatrix} l_xA \\ l_yA \\ l_xB \\ l_yB \end{bmatrix}$$

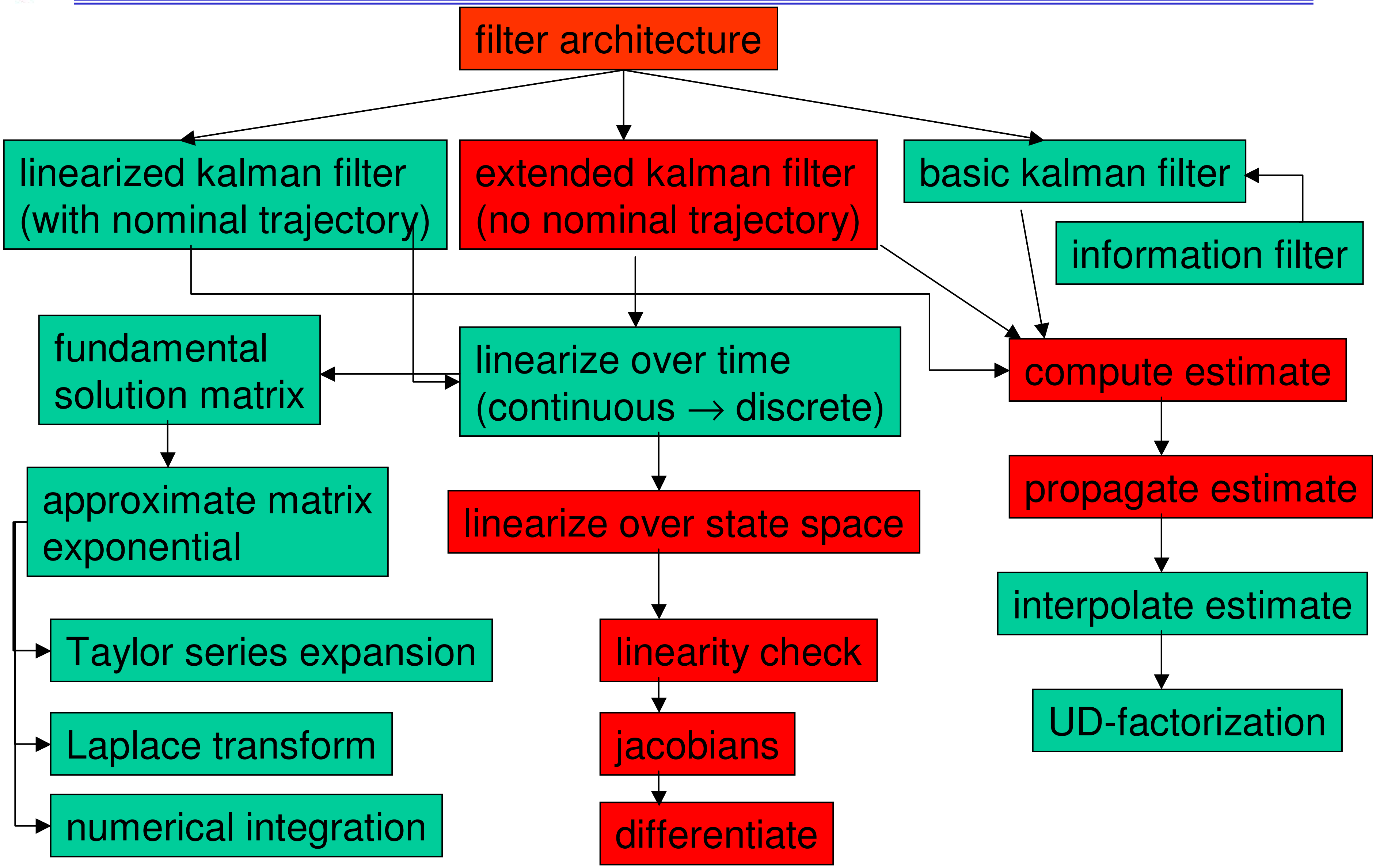
measurement noise

$$\begin{bmatrix} os.x.A.RAW \\ y.A \\ x.B \\ y.B \end{bmatrix} = \begin{bmatrix} [CAL.A.INV]\cdot\left(\begin{bmatrix} p.x.i \\ p.y.i \end{bmatrix} + [ROT]\cdot\begin{bmatrix} l_xA \\ l_yA \end{bmatrix} - [B.A]\right) \\ [CAL.B.INV]\cdot\left(\begin{bmatrix} p.x.i \\ p.y.i \end{bmatrix} + [ROT]\cdot\begin{bmatrix} l_xB \\ l_yB \end{bmatrix} - [B.B]\right) \end{bmatrix} + \begin{bmatrix} \epsilon_{xA} \\ \epsilon_{yA} \\ \epsilon_{xB} \\ \epsilon_{yB} \end{bmatrix}$$

measurement variables



# Kalman Filter Synthesis



# FY03: Assumption-based Program Synthesis

- explicitly call out assumptions in:
  - model, e.g., CM of vehicle = geometric center
  - synthesis process, e.g., is Taylor series approx. appropriate?
- track assumptions during synthesis
  - document assumptions in code
  - consistency check on assumptions
  - generate runtime monitors for assumptions
  - generate alternative programs based on different assumptions, e.g., particle filter vs EKF

- Synthesis of code for multiple control modes and sensor configurations
  - ACS detumble, IMU + star-tracker, IMU only